Definition

```
def LoadImmediate(op,rd,rc,offset) as
    case op of
        LI16L, LI32L, LI8, LI16AL, LI32AL, LI16B, LI32B, LI16AB, LI32AB:
        LI64L, LI64AL, LI64B, LI64AB:
            signed ← true
        LIU16L, LIU32L, LIU8, LIU16AL, LIU32AL,
        LIU16B, LIU32B, LIU16AB, LIU32AB:
        LIU64L, LIU64AL, LIU64B, LIU64AB:
            signed ← false
        LI128L, LI128AL, LI128B, LI128AB:
            signed ← undefined
    endcase
    case op of
        LI8, LIU8:
            size ← 8
        LI16L, LIU16L, LI16AL, LIU16AL, LI16B, LIU16B, LI16AB, LIU16AB:
            size ← 16
        LI32L, LIU32L, LI32AL, LIU32AL, LI32B, LIU32B, LI32AB, LIU32AB:
            size ← 32
        LI64L, LIU64L, LI64AL, LIU64AL, LI64B, LIU64B, LI64AB, LIU64AB:
            size ← 64
        LI128L, LI128AL, LI128B, LI128AB:
            size ← 128
    endcase
    lsize ← log(size)
    case op of
        LI16L, LIU16L, LI32L, LIU32L, LI64L, LIU64L, LI128L,
        LI16AL, LIU16AL, LI32AL, LIU32AL, LI64AL, LIU64AL, LI128AL:
            order ← LI
        LI16B, LIU16B, LI32B, LIU32B, LI64B, LIU64B, LI128B,
        LI16AB, LIU16AB, LI32AB, LIU32AB, LI64AB, LIU64AB, LI128AB:
            order ← B
        LI8, LIU8:
            order ← undefined
    endcase
```

Fig. 51C

```
    c ← RegRead(rc, 64)
    VirtAddr ← c + (offset[55-lsize] || offset || 0[lsize-3])
    case op of
        LI16AL, LIU16AL, LI32AL, LIU32AL, LI64AL, LIU64AL, LI128AL,
        LI16AB, LIU16AB, LI32AB, LIU32AB, LI64AB, LIU64AB, LI128AB:
            if (c[lsize-4..0] ≠ 0 then
                raise AccessDisallowedByVirtualAddress
            endif
        LI16L, LIU16L, LI32L, LIU32L, LI64L, LIU64L, LI128L,
        LI16B, LIU16B, LI32B, LIU32B, LI64B, LIU64B, LI128B:
        LI8, LIU8:
    endcase
    m ← LoadMemory(c,VirtAddr,size,order)
    a ← (m[size-1] and signed)[128-size] || m
    RegWrite(rd, 128, a)
enddef
```

Exceptions

Access disallowed by virtual address
Access disallowed by tag
Access disallowed by global TB
Access disallowed by local TB
Access detail required by tag
Access detail required by local TB
Access detail required by global TB
Local TB miss
Global TB miss

Fig. 51C (cont)

Operation codes

S.8	Store byte
S.16.B	Store double big-endian
S.16.A.B	Store double aligned big-endian
S.16.L	Store double little-endian
S.16.A.L	Store double aligned little-endian
S.32.B	Store quadlet big-endian
S.32.A.B	Store quadlet aligned big-endian
S.32.L	Store quadlet little-endian
S.32.A.L	Store quadlet aligned little-endian
S.64.B	Store octlet big-endian
S.64.A.B	Store octlet aligned big-endian
S.64.L	Store octlet little-endian
S.64.A.L	Store octlet aligned little-endian
S.128.B	Store hexlet big-endian
S.128.A.B	Store hexlet aligned big-endian
S.128.L	Store hexlet little-endian
S.128.A.L	Store hexlet aligned little-endian
S.MUX.64.A.B	Store multiplex octlet aligned big-endian
S.MUX.64.A.L	Store multiplex octlet aligned little-endian

Fig. 52A

Selection

number format	op	size	alignment	ordering
byte		8		
integer		16 32 64 128		L B
integer aligned		16 32 64 128	A	L B
multiplex	MUX	64	A	L B

Format

op rd,rc,rb

op(rd,rc,rb)

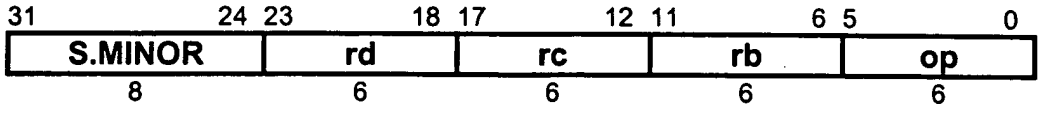

Fig. 52B

Definition

```
def Store(op,rd,rc,rb) as
    case op of
        S8:
            size ← 8
        S16L, S16AL, S16B, S16AB:
            size ← 16
        S32L, S32AL, S32B, S32AB:
            size ← 32
        S64L, S64AL, S64B, S64AB,
        SMUX64AB, SMUX64AL:
            size ← 64
        S128L, S128AL, S128B, S128AB:
            size ← 128
    endcase
    lsize ← log(size)
    case op of
        S8:
            order ← undefined
        S16L, S32L, S64L, S128L,
        S16AL, S32AL, S64AL, S128AL, SMUX64ALI:
            order ← L
        S16B, S32B, S64B, S128B,
        S16AB, S32AB, S64AB, S128AB, SMUX64ABI:
            order ← B
    endcase
    c ← RegRead(rc, 64)
    b ← RegRead(rb, 64)
    VirtAddr ← c + (b66-lsize..0 || 0lsize-3)
    case op of
        S16AL, S32AL, S64AL, S128AL,
        S16AB, S32AB, S64AB, S128AB,
        SMUX64AB, SMUX64AL:
            if (clsize-4..0 ≠ 0 then
                raise AccessDisallowedByVirtualAddress
            endif
        S16L, S32L, S64L, S128L,
        S16B, S32B, S64B, S128B:
        S8:
    endcase
```

Fig. 52C

```
    d ← RegRead(rd, 128)
    case op of
        S8,
        S16L, S16AL, S16B, S16AB,
        S32L, S32AL, S32B, S32AB,
        S64L, S64AL, S64B, S64AB,
        S128L, S128AL, S128B, S128AB:
            StoreMemory(c,VirtAddr,size,order,d_size-1..0)
        SMUX64AB, SMUX64AL:
            lock
                a ← LoadMemoryW(c,VirtAddr,size,order)
                m ← (d_127..64 & d_63..0) | (a & ~d_63..0)
                StoreMemory(c,VirtAddr,size,order,m)
            endlock
    endcase
enddef
```

Exceptions

Access disallowed by virtual address
Access disallowed by tag
Access disallowed by global TB
Access disallowed by local TB
Access detail required by tag
Access detail required by local TB
Access detail required by global TB
Local TB miss
Global TB miss

Fig. 52C (cont)

Operation codes

S.I.8	Store immediate byte
S.I.16.A.B	Store immediate double aligned big-endian
S.I.16.B	Store immediate double big-endian
S.I.16.A.L	Store immediate double aligned little-endian
S.I.16.L	Store immediate double little-endian
S.I.32.A.B	Store immediate quadlet aligned big-endian
S.I.32.B	Store immediate quadlet big-endian
S.I.32.A.L	Store immediate quadlet aligned little-endian
S.I.32.L	Store immediate quadlet little-endian
S.I.64.A.B	Store immediate octlet aligned big-endian
S.I.64.B	Store immediate octlet big-endian
S.I.64.A.L	Store immediate octlet aligned little-endian
S.I.64.L	Store immediate octlet little-endian
S.I.128.A.B	Store immediate hexlet aligned big-endian
S.I.128.B	Store immediate hexlet big-endian
S.I.128.A.L	Store immediate hexlet aligned little-endian
S.I.128.L	Store immediate hexlet little-endian
S.MUXI.64.A.B	Store multiplex immediate octlet aligned big-endian
S.MUXI.64.A.L	Store multiplex immediate octlet aligned little-endian

Fig. 53A

Selection

number format	op	size	alignment	ordering
byte		8		
integer		16 32 64 128		L B
integer aligned		16 32 64 128	A	L B
multiplex	MUX	64	A	L B

Format

S.op.l.size.align.order rd,rc,offset

sopisizealignorder(rd,rc,offset)

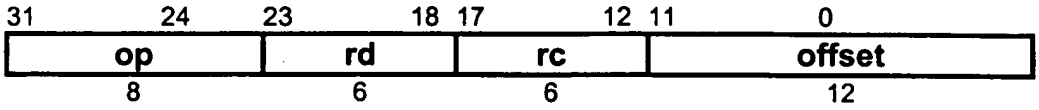

Fig. 53B

Definition

```
def StoreImmediate(op,rd,rc,offset) as
    case op of
        SI8:
            size ← 8
        SI16L, SI16AL, SI16B, SI16AB:
            size ← 16
        SI32L, SI32AL, SI32B, SI32AB:
            size ← 32
        SI64L, SI64AL, SI64B, SI64AB, SMUXI64AB, SMUXI64AL:
            size ← 64
        SI128L, SI128AL, SI128B, SI128AB:
            size ← 128
    endcase
    lsize ← log(size)
    case op of
        SI8:
            order ← undefined
        SI16L, SI32L, SI64L, SI128L,
        SI16AL, SI32AL, SI64AL, SI128AL, SMUXI64AL:
            order ← L
        SI16B, SI32B, SI64B, SI128B,
        SI16AB, SI32AB, SI64AB, SI128AB, SMUXI64AB:
            order ← B
    endcase
    c ← RegRead(rc, 64)
    VirtAddr ← c + (offset[55]^(55-lsize) || offset || 0^(lsize-3))
    case op of
        SI16AL, SI32AL, SI64AL, SI128AL,
        SI16AB, SI32AB, SI64AB, SI128AB,
        SMUXI64AB, SMUXI64AL:
            if (c[lsize-4..0] ≠ 0 then
                raise AccessDisallowedByVirtualAddress
            endif
        SI16L, SI32L, SI64L, SI128L,
        SI16B, SI32B, SI64B, SI128B:
        SI8:
    endcase
```

Fig. 53C

```
    d ← RegRead(rd, 128)
    case op of
        SI8,
        SI16L, SI16AL, SI16B, SI16AB,
        SI32L, SI32AL, SI32B, SI32AB,
        SI64L, SI64AL, SI64B, SI64AB,
        SI128L, SI128AL, SI128B, SI128AB:
            StoreMemory(c,VirtAddr,size,order,d_size-1..0)
        SMUXI64AB, SMUXI64AL:
            lock
                a ← LoadMemoryW(c,VirtAddr,size,order)
                m ← (d_127..64 & d_63..0) | (a & ~d_63..0)
                StoreMemory(c,VirtAddr,size,order,m)
            endlock
    endcase
enddef
```

Exceptions

Access disallowed by virtual address
Access disallowed by tag
Access disallowed by global TB
Access disallowed by local TB
Access detail required by tag
Access detail required by local TB
Access detail required by global TB
Local TB miss
Global TB miss

Fig. 53C (cont)